Capitolo 6

OPERAZIONI I/O SU FILE

Il C++ supporta il sistema di Input/Output del C, ma dispone anche di un proprio sistema ad oggetti molto complesso e potente, del quale nel seguito si vedranno solo alcune delle caratteristiche base. Per approfondire e sfruttare al meglio le possibilità offerte in questo campo dal C++ fare riferimento a qualunque manuale del linguaggio.

Per leggere o scrivere su file, in C++ bisogna aprire uno **stream** (= flusso) ed associarlo al file. Uno stream è un dispositivo logico che fa da interfaccia al dispositivo fisico a cui è associato. Gli stream sono largamente indipendenti dal dispositivo fisico, per cui le stesse funzioni di I/O possono essere utilizzate per dispositivi fisici diversi: per esempio per scrivere su disco, nastro, schermo, stampante. Esistono tre tipi di stream: di input, di output e di input/output.

All'inizio dell'esecuzione di un programma in C++ vengono aperti automaticamente alcuni stream predefiniti. I più importanti sono *cin* e *cout*, collegati rispettivamente alla tastiera e allo schermo.

Per creare ed eseguire operazioni su uno stream si deve includere nel programma l'intestazione *fstream.h*:

```
#include <fstream.h>
```

Fstream.h contiene la definizione di numerose classi, tra le quali le seguenti sono da usare per creare un oggetto stream:

ifstream $\longmapsto$ operazioni di input

ofstream $\longmapsto$ operazioni di output

fstream $\longmapsto$ operazioni di input e output

Il costruttore della classe ofstream crea lo stream di output ed apre il file indicato (o lo crea se non esiste) in modalità scrittura cancellandone il contenuto preesistente.

Ifstream apre il file (o lo crea se non esiste) in modalità lettura, posizionandosi all'inizio.

Fstream apre il file (o lo crea se non esiste) in modalità lettura e scrittura, posizionandosi all'inizio.

Generalmente è opportuno controllare immediatamente se l'operazione di creazione stream e apertura file è andata a buon fine.

Esempio: creazione dello stream di input *miostream* per leggere il contenuto del file *miofile*:

```
ifstream miostream("miofile");
if (!miostream)

    {
    cout << "Impossibile aprire il file !" << endl;
    return 1;
    }

...
```

Le operazioni di lettura e scrittura su un file di testo si possono eseguire molto semplicemente con gli operatori << ed >>. Il seguente spezzone di programma apre il file *miofile* in scrittura, vi inserisce tre righe di caratteri prelevati da tastiera e lo chiude. Quindi lo riapre in sola lettura e legge riga per riga il contenuto fino alla fine del file, mandandolo in uscita sullo schermo. Notare l'uso di eof(), una funzione membro di ifstream, ofstream ed fstream.

```
ofstream miostream("miofile");
if (!miostream)

    {
    cout << "Impossibile aprire il file !";
    return 1;
    }

for(i=1; i<= 3; i++)

    {
    cin >> temp;
    miostream << temp << endl;
    }

miostream.close();

ifstream miostream("miofile");
if(!miostream)

    {
    cout << "Impossibile aprire il file !";
```

```
        return 1;
        }

    while(!miostream.eof())

        {
        miostream >> temp;
        cout << temp << endl;
        }

    miostream.close();
    . . .
```

Nella lettura di un file di testo con l'operatore >> avvengono alcune traduzioni di caratteri. Per evitare ciò bisogna aprire il file in modalità binaria ed utilizzare le opportune funzioni, per esempio get() e put():

istream &get(char &*carattere*)

ostream &put(char *carattere*)

La funzione get() legge un singolo carattere dallo stream chiamante e ne inserisce il valore in *carattere*, restituendo l'indirizzo dello stream. La funzione put() scrive *carattere* sullo stream e restituisce l'indirizzo dello stream.

Ecco un esempio in cui viene aperto un file in modalità binaria e con l'uso della funzione get() viene letto il contenuto del file:

```
    #include <iostream.h>
    #include <fstream.h>

    int main( int argc, char *argv[ ] )

        {
        char ch;
        ifstream pippo(argv[1], ios::in | ios::binary);
        if (!pippo)
            {
            cout << "File non aperto";
            return 1;
            }
        while(pippo)
            {
            pippo.get(ch);
            if(pippo)
                cout << ch;
```

```
        }
    return 0;
    }
```

Nota: pippo è uguale a zero quando viene raggiunta la fine del file.

CONTINUA ...